AVR32752: Using the AVR32 UC3 Static Memory Controller

Features

- **Several Types of Access Supported**
  - **8-bit Access Mode**
  - **16-bit Access Mode**
- **Software Configurable**
  - **Timing Parameters**
  - **Initializations**

1 Introduction

The AVR32® UC3 microcontroller has a dedicated Static Memory Controller (SMC). This controller is highly flexible and capable of supporting a series of external static memory devices. Timing values and parameters are calculated from the external static memory device's datasheet and programmed into the SMC controller. This is also true for various types of access with respect to read and write operations. When the controller is correctly instantiated, the external device can then be accessed as a normal memory mapped device. The controller provides a seamless and transparent interface to a memory mapped SMC device.

The purpose of this application note is to understand all features available in the SMC and to deliver guidelines to interface AVR32 UC3 microcontroller and external static memory devices.

32-bit AVR® Microcontrollers

Application Note

Rev. 7813B-AVR32-01/09

2 Abbreviations

- **EBI**: External Bus Interface
- **GPIO**: General Purpose Input Output
- **HMATRIX**: HSB Bus Matrix
- **HSB:** High Speed Bus
- **PBA**: Peripheral Bus A
- **PBB**: Peripheral Bus B
- **SMC**: Static Memory Controller

3 SMC Features

Several features are available in the SMC:

- Chip select.
- Programmable pulse, setup and hold times for read and write accesses per chip select.
- 8- and16-bit Data Bus.

The SMC is capable of 8-bit or 16-bit data path, and supports byte, half-word and word access. Bursts are supported for both write and read access.

3.1 Device characteristics

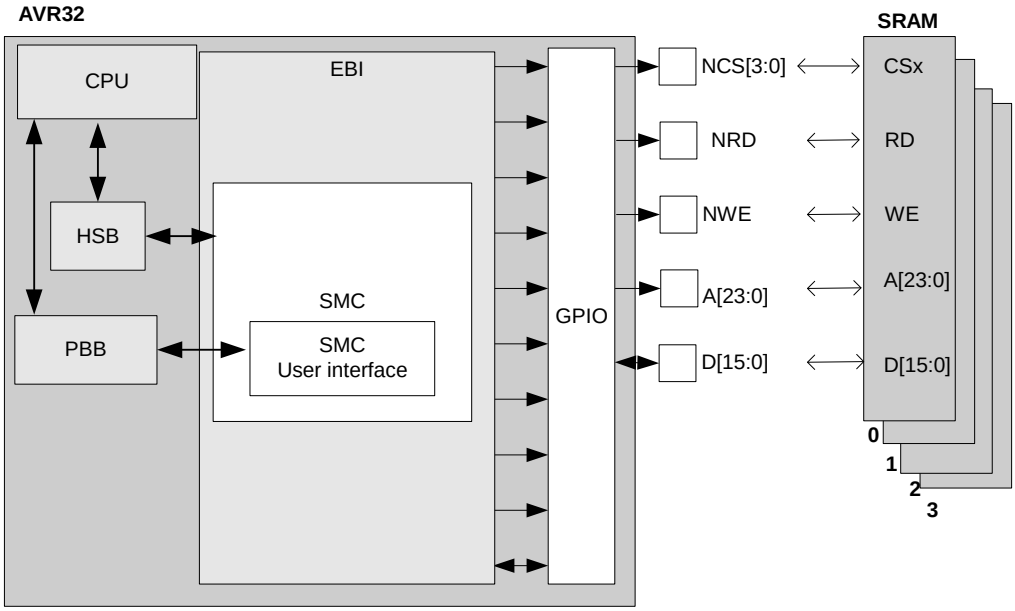
The SMC is able to drive four different static memory devices. Each static memory device has specific timings and command sequence. These characteristics are stored into the SMC to control the data flow between the AVR32 and the static memory device. Once the SMC is configured, the access to the static memory device is seamless to the user. Each chip select is assigned to a memory area (check the “Physical Memory Map” section of the datasheet).

3.2 Access

The SMC Controller is part of the EBI (External Bus Interface), which is accessed through the System Bus from the CPU core. The SMC Controller may connect to an external static memory device through the GPIO. A conceptual schematic of the path is illustrated in Figure 3-1.

3.3 Physical Layout and Access

Figure 3-1. Interface with a 16-bit Static Memory Device



Multiple static memory devices can be connected together to the SMC to increase the amount of static memory devices available to the system.

4 SMC Controller

The SMC must be configured before starting to communicate with static memory device.

4.1 Source files

The configuration of the SMC is accessible through the software driver contained in the AVR32752.zip file delivered with the application note and available on www.atmel.com.The source file location of Table 4-1 is related to the following location src/DRIVERS/EBI/SMC/.

Table 4-1. Driver file names

Source file	Description
smc.c/smc.h	SMC Driver: - Setup of Alternate Functions Used - Setup of SMC Mode Used - Setup of Timing Used
conf_ebi.h	Define the pin mapping for the application. Here, all alternate pins used should be defined

	Customize this file for the application.
	Enable the chipselect required using the definition:
	`#define USE_NCSx  //x :  the number of the chipselect`
	`#define COMPONENT_CSx  "smc_peripheral.h"`
	Duplicate these two definitions if the application requires more than 1 SMC device.

smc_peripheral.h	Template to address SMC device component. Duplicate and rename if the application requires more than 1 SMC device. It contains : - Memory Size Definition. - Data Bus Width Definition. - Address Length Definition. - Mode Definition.

4.2 Initialization

After a reset, the AVR32 is not configured to access an external static memory device. The sequence for initializing the SMC after a successful reset should be as follows:

- Initialize the SMC and timing characteristics for the device (see smc.c file).

```
// Setup SMC for NCS0
SMC_CS_SETUP(0)
```

- Configure the GPIO controller (see smc.c file).

```
// Setup Alternate function for SMC
gpio_enable_module(SMC_EBI_GPIO_MAP, sizeof(SMC_EBI_GPIO_MAP)
/ sizeof(SMC_EBI_GPIO_MAP[0]));
```

4.3 Data Bus Width Definition

A data bus width of 8 or 16 bits can be selected for each chip select. This option is controlled by the field DBW in MODE (Mode Register) for the corresponding chip select (see `smc_peripheral.h` file):

```
//! SMC Data Bus Width
#define SMC_DBW                 16
```

4.4 Mode Definition

Each chip select with a 16-bit data bus can operate with one of two different types of write access: byte write or byte select access. This is controlled by the BAT (BAT = Byte Access Type) field of the MODE register for the corresponding chip select. Byte write access supports one byte write signal per byte of the data bus and a single read signal. (see `smc_peripheral.h` file):

```
//! Whether 8-bit SM chips are connected on the SMC
#define SMC_8_BIT_CHIPS             TRUE   //Byte Write Access
#define SMC_8_BIT_CHIPS             FALSE  //Byte Select Access
```

In byte select access mode, read/write operations can be enabled/ disabled at a byte level. One byte-select line per byte of the data bus is provided. One NRD and one NWE signal control read and write operations.

5 Analysis for Read-Write Access

In order to better understand all the signals exchanged between the SMC and static memory device, here are some examples of usage of the SMC software driver with different test cases of hardware interfaces.

5.1 Source code used for analysis

Each analysis differs from the others by changes in the `smc_peripheral.h` file and by the data bus width field (DBW).

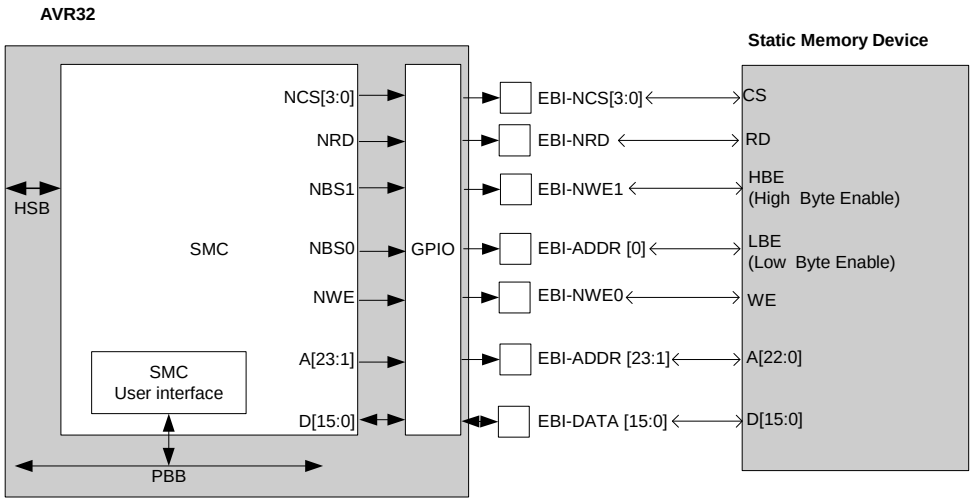
5.2 16-bit Mode (DBW='1')

In this first test case, the SMC is setup with a data-bus width of 16 bits. We will analyze this first connection with different variable width access (16-bit and 8-bit data access).

5.2.1 Writing 16-bit data into a 16-bit static memory device

The data bus width is configured to the maximum size available (16) and the byte access mode should be configured to mode 0 (BAT='0'). In that case, the SMC provides two byte select lines NBS0 and NBS1. NBS0 is used to access the lower byte of the static memory device byte and NBS1 is used to access the higher byte of the static memory device.

Figure 5-1. Interface with a 16-bit Static Memory Device



Refer to `smc_peripheral.h`

```
//! SMC Data Bus Width
#define SMC_DBW                 16
//! Whether 8-bit SM chips are connected on the SMC
#define SMC_8_BIT_CHIPS         FALSE
```

To perform a write instruction:

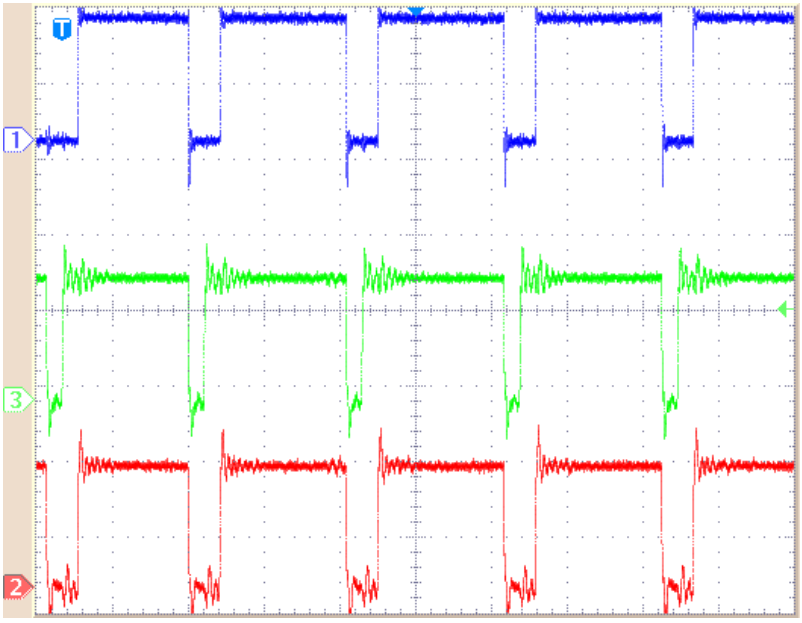
```
//! SMC Write Macro Function in 16-bit mode.
#define smc_write_16(b,o,d) (((volatile U16 *)(b))[(o)] = (U16)(d))
smc_write_16(SRAM,i,i);
```

NBS0 will toggle once, for data access as

Figure 5-2 shows.

Figure 5-2. DBW='1', BAT='0' and 16-bits write access.

- **Channel 1** – **NBS0**-
- **Channel 3** – **NWE0**
- **Channel 2** - **NBS1**-

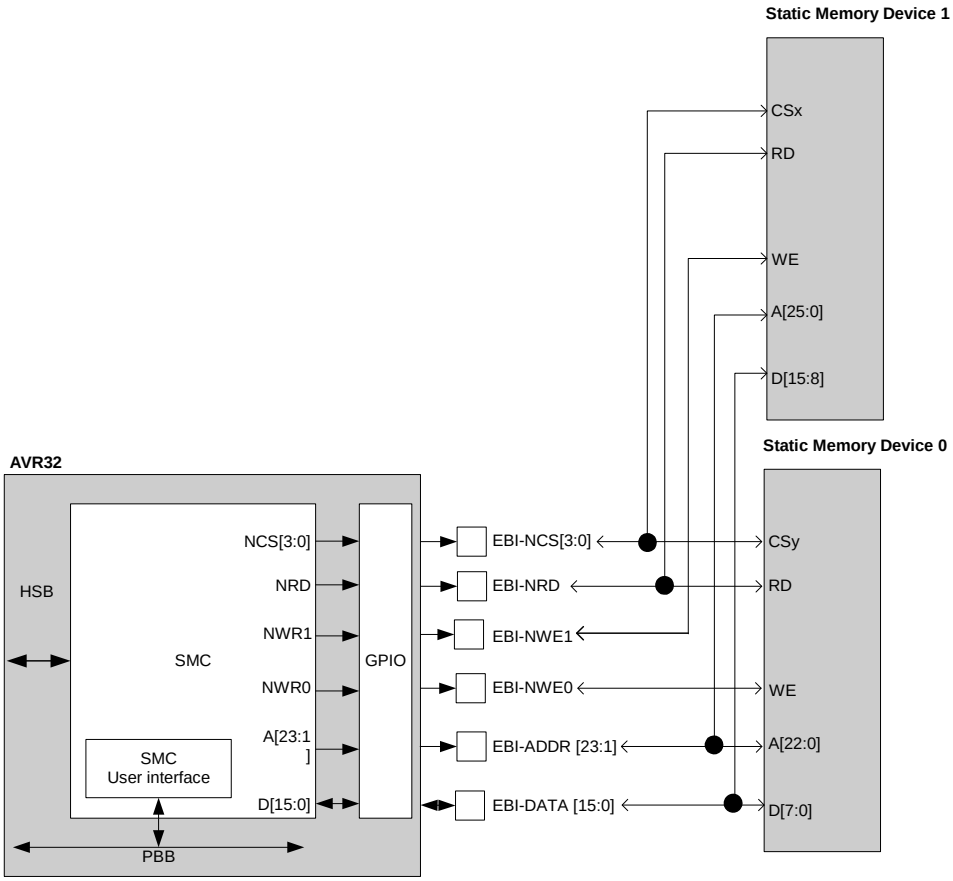


5.2.2 Writing 16-bit data into two 8-bit data static memory devices

The data bus width is configured to the maximum size available (16) and in case of 2x8-bit devices access, the byte access should be configured to mode 1 (BAT='1'). In that case, the SMC provides two write enable lines NWR0 and NWR1.

NWR0 and NWR1 are used for respectively access to the static memory device 0 (lower byte D7-D0) and static memory device 1 (higher byte D15-D8).

Figure 5-3. Interface with 2x 8-bit Static Memory Devices



Refer to `smc_peripheral.h`

```
…
//! SMC Data Bus Width
#define SMC_DBW                 16
//! Whether 8-bit SM chips are connected on the SMC
#define SMC_8_BIT_CHIPS         TRUE
```

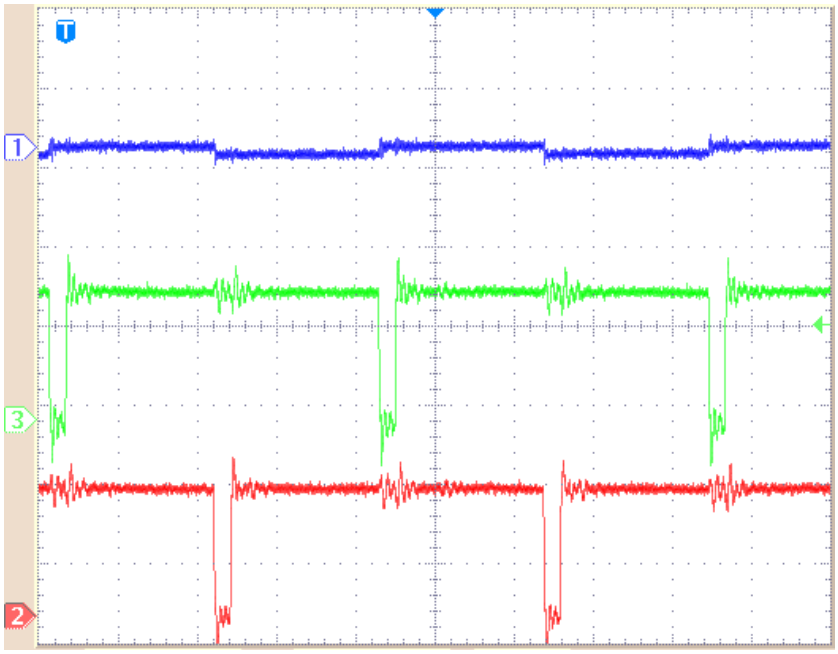
To perform the write instruction:

```
//! SMC Write Macro Function in 16-bit mode.
#define smc_write_16(b,o,d) (((volatile U16 *)(b))[(o)] = (U16)(d))
smc_write_16(SRAM,i,i);
```

NWR0 will toggle for event address access and NWR1 will toggle for odd address access, as **Figure 5-4** shows.

Figure 5-4. DBW='1', BAT='1' and 8-bits write access.

- **Channel 1** – **A0 line**
- **Channel 3** – **NWR0**
- **Channel 2** – **NWR1**

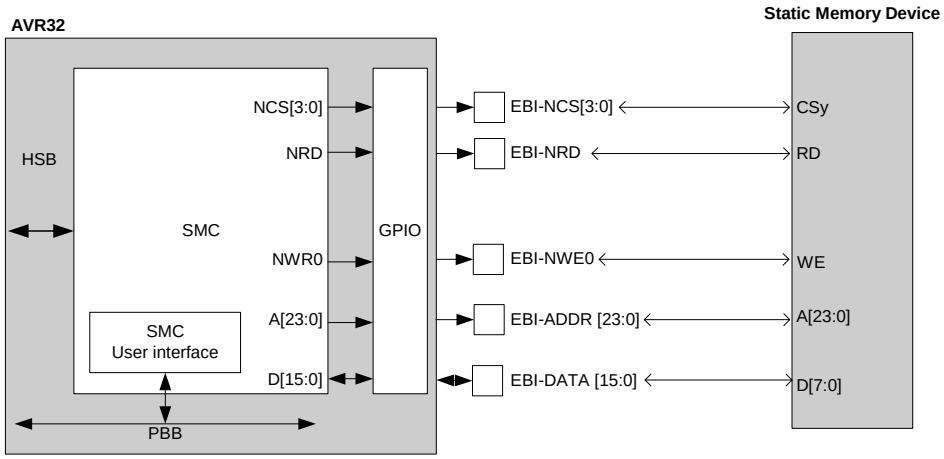


5.3 8-bit Mode (DBW='0')

The SMC Controller is setup with a data-bus width of 8 bits.

The data bus width is configured to the minimum size available, 8, and there is no dependency to the byte access mode (BAT='0'/'1'). In that case, the SMC provides one write enable line for write access to the device.

Figure 5-5. Interface with an 8-bit Static Memory Device



Refer to `smc_peripheral.h`

```
//! SMC Data Bus Width
#define SMC_DBW                 8
//! Whether 8-bit SM chips are connected on the SMC
#define SMC_8_BIT_CHIPS         TRUE/FALSE (Ignored)
```

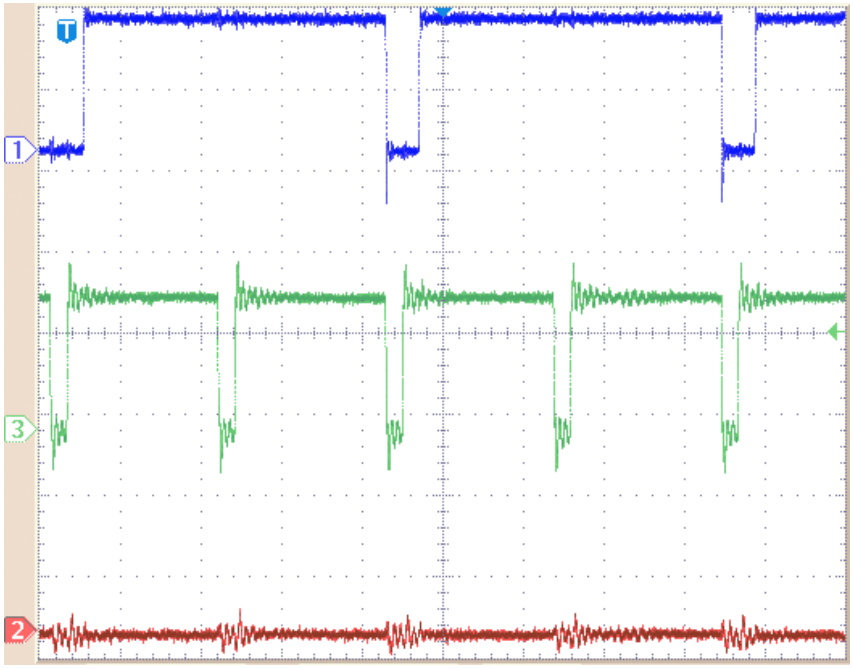
To perform the write instruction:

```
//! SMC Write Macro Function in 8-bit mode.
#define smc_write_8(b,o,d) (((volatile U8 *)(b))[(o)] = (U8)(d))
smc_write_8(SRAM,i,i);
```

NWR0 will toggle twice, for high to low data access as **Figure 5-6** shows.

Figure 5-6. DBW='0' and 8-bits write access.

- **Channel 1** – **A0 line**
- **Channel 3** – **NWR0**
- **Channel 2** – **NWR1**



Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
Avr32@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature